НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

УДК: 004.051

«До захисту допущено»
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“___” _____2019 р.

Дипломний проект
на здобуття ступеня бакалавра

з напряму підготовки *6.050101 «Комп’ютерні науки»*

на тему: « *Інформаційно-аналітична система пошуку продуктів* »

Виконав: студент *4* курсу, групи *ІС-з51*
Ткаченко Роман Олександрович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *старший викладач Новікова П.А.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації *доц., к.т.н., доц. Тєлишева Т.О.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент *старший викладач каф. ОТ Алещенко О.В.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент Ткаченко Р.О. _____
(підпис)

Київ – 2019 р.

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проекту складається з п’яти розділів, містить 18 рисунків, 7 таблиць, 1 додаток, 28 джерел.

У дипломному проекті реалізована тема «Інформаційно-аналітична система пошуку продуктів», метою якої є спрощення пошуку продуктів, завдяки: - збереженню та упорядкуванню велокої кількості продуктів; - економії часу та ресурсів; - покращення якості пошуку.

Розділ «Загальні положення» описує процес діяльності та можливі варіанти використання даної системи. В цьому розділі розглядаються та аналізуються аналоги даної системи, задається ціль та мета розробки, а також задачі, які треба реалізувати.

В розділі «Інформаційне забезпечення» описана структура вхідих та вихідних даних, наведена структура бази даних.

В розділі «Математичне забезпечення» наведено детальний опис математичних алгоритмів, що були використані.

В розділі «Програмне та технічне забезпечення» описані основні програмні засоби, які були використані для розробки даної програми, наведені технічні вимоги до системи, на якій буде запускатися програма, описана програмна архітектура, яка була обрана для розробки.

ПОШУК, ПРОДУКТИ, ТОВАРИ, МАГАЗИНИ, КОМПАНІЇ, ВІДГУКИ.

SUMMARY

Structure and scope of work. Explanatory note of the diploma project consists of five sections, containing 18 figures, 7 tables, 1 supplement, 28 sources.

The diploma project implemented the theme “ Information and analytical system of product search”, which aims to simplify the search for products , thanks to: - the preservation and ordering of a large number of products; - saving time and sources; - improved search quality.

The section “General” describes the process of activity and possible options for using this system. This section looks at and analyses the analogs of this system, defines the purpose and purpose of the development, a well as the tasks that need to be implemented.

In the ection “Information support” decribes the structure of imput and output data, the structure of the database is presented.

The section “Mathematival support” provides a detailed description of the mathematical algorithms that have been used.

The “Software and Hardware” section describes the main software tools that were used to develop this program, lists the technical requirements for the system on which the program will be launched, describes the software architecture that has been selected for development.

SEARCH, PRODUCTS, STORES, COMPANIES, REVIEWS, WEB-APPLICATION

ВСТУП

Сьогодення потребує швидкого реагування на зміни у житті та величезне значення приділяє економії часу при пошуку будь-якої інформації, документа, товару, споживчого продукту. Особливе місце у житті кожної людини, яка працює та багато часу приділяє роботі, займають повсякденні речі. Тому при обранні цієї теми, постало дуже важливе та актуальне питання - не витрачати зайвого часу на пошуки потрібних речей, продуктів, споживчих товарів. Можливість найшвидшого, найякіснішого та найближчого за геолокацією необхідного продукту стане в нагоді усім, хто цінує свій час.

Пошукова система - це складний програмно-апаратний комплекс, який призначений для здійснення пошуку ресурсів в Інтернет, збереження відомостей про них в своїх базах і надання користувачу переліку посилань відповідно до його пошукового запиту. Головним завданням пошукової системи є здатність надавати користувачам саме ту інформацію, яку вони шукають. [1]

Самі пошукові системи з'явилися в грудні 1990 року. Пошук «Хто є користувачем» датується 1982, а багаторазовий пошук користувачів інформаційної служби Knowbot був вперше здійснений у 1989 році. Першою добре документованою пошуковою системою, яка шукала файли вмісту, а саме файли FTP, був Арчі, який дебютував 10 вересня 1990 року. [1]

До вересня 1993 року Всесвітня мережа була повністю індексована вручну. Існував список веб-серверів, які редагував Тім Бернерс-Лі і розміщувався на веб-сервері CERN. Один знімок списку в 1992 році залишається, але, оскільки все більше і більше веб-серверів виходили в Інтернет, центральний список більше не міг встигати. [2]

Інформаційно - аналітична система — це система, яка дозволяє отримувати інформацію, створювати її та здійснювати її обробку та аналіз. Її завданням є ефективне зберігання, обробка та аналіз даних. [3]

Робити «чіткі» запити до пошукової системи, які відповідають її принципам роботи неможливо. Тому, розробники створюють такі алгоритми і принципи роботи пошукових систем, які найкраще пристосовані до поведінки і ходу думок пересічного користувача. Пошукова система повинна діяти так само, як діє користувач при пошуку інформації і надавати за його запитом інформацію максимально швидко і просто.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

Місією інформаційно-аналітично системи є збір, переробка та аналіз інформації, потрібної для ефективного управління ресурсами, створення інформаційного та технічного середовища для управління її діяльністю.

Одним з таких корисних «помічників» як для власників сервісів швидкої доставки продуктів, так і для пересічних громадян стане в нагоді пошукова інформаційно-аналітична система продуктів, яка має на меті не тільки пошук необхідного, конкретного типу продукту, а ще й найближчого за геолокацією. Це дозволить економити не тільки час, а й деякі матеріальні ресурси та транспортні витрати.

Даний проект базується на необхідності створенні пошукової системи найближчого за знаходженням конкретного споживчого продукту для економії часу та матеріальних ресурсів.

1.1.1 Опис процесу діяльності

На рисунку 1.1 зображено діаграму діяльності сайту

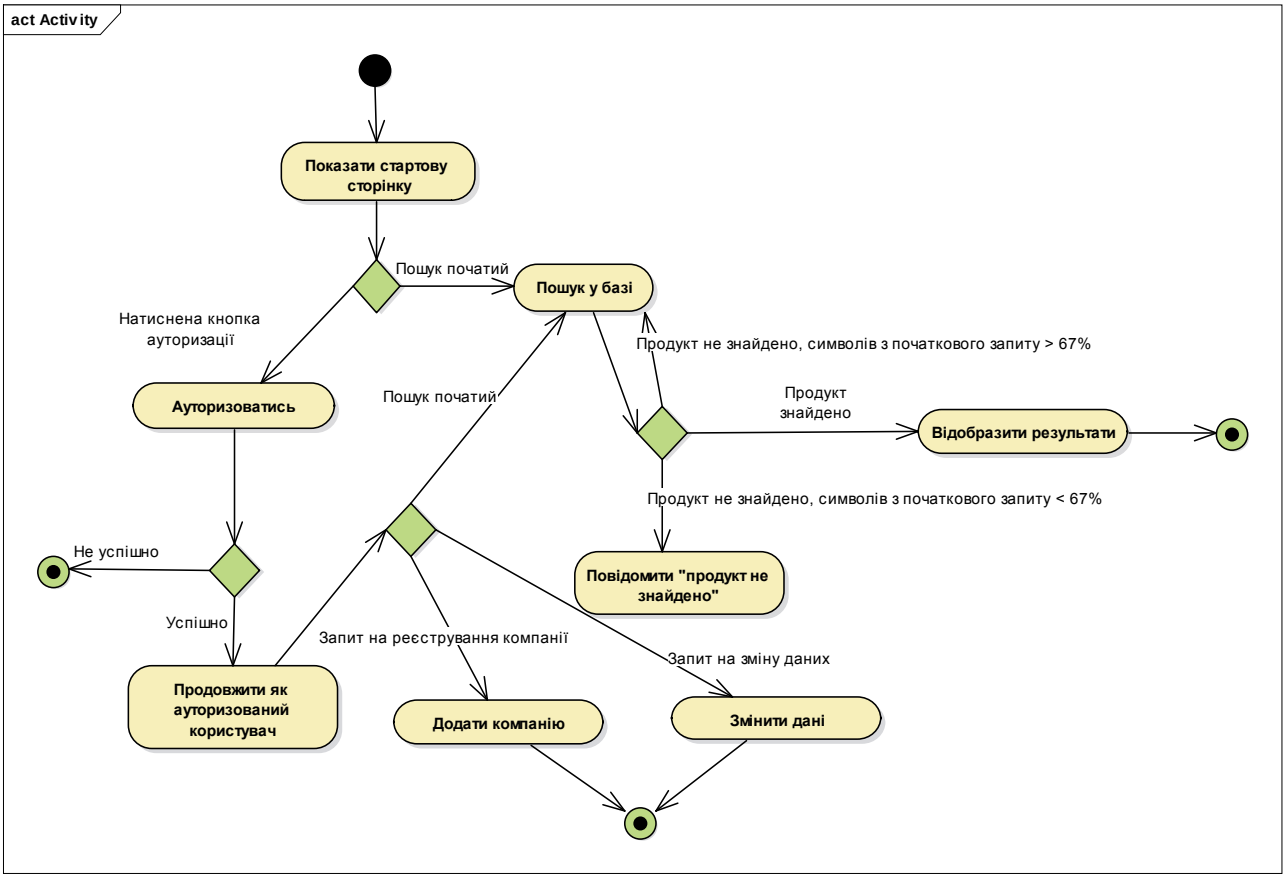

Рисунок 1.1 – Діаграма діяльності

1.1.2 Опис функціональної моделі

Дійові особи:

Неавторизований відвідувач – має можливості зареєструватись або авторизуватись через Facebook та Google Account, виконати пошук та сортування продукту за назвою через строку пошуку або вибрати з популярних категорій та підкатегорій, переглянути відгуки по продукту інших користувачів та прокласти маршрут до вибраної точки продажу товару на мапі.

Авторизований відвідувач – має всі фукціональні можливості неавторизованого відвідувача та можливості залишати відгуки до конкретного товару, магазину. Авторизований користувач також має можливість реєструвати власну точку продажу товару (магазин) і додавати асортимент через спеціальну форму.

Модератор – окрім можливостей авторизованого користувача проводить модерацію відгуків, наповнення магазинів та перевіряє валідність асортименту нових точок продажу.

1.2 Огляд наявних аналогів

Частково схожі функції (пошуку ціни товару) мають сайти великих продуктових та будівельних компаній, таких як: metro.ua, epicentrk.ua

Частково схожі функції (пошуку ціни і відгуків про товар) мають тематичні мобільні додатки.

Найбільш схожими є функції сайту Prom.ua.

Спільне: пошук та розміщення продукту та компаній, система відгуків

Відрізняється (аналог від поточного проекту): орієнтовністю на оптовий продаж та доставлення поштою, частковою відсутністю конкретного адресу розташування точок продажу та сортування пропозицій по віддаленості від користувача.

1.3 Постановка задачі

Спроектувати та розробити інформаційно-аналітичну систему швидкого пошуку продуктів за геолокацією з можливістю реєстрації компаній та магазинів. Передбачити різні можливості додавання даних.

1.3.1 Призначення розробки

Призначенням розробки є організація та ефективне використання часу, матеріальних ресурсів для набуття економічних вигод в майбутньому.

1.3.2 Цілі та задачі розробки

Метою розробки дипломного проекту є спрощення і підвищення якості пошуку різноманітних товарів.

Задачами розробки дипломного проекту є:

- Розробка пошукової системи
- Ауторизація через сервіси Facebook та Google Accounts
- Сортування за віддаленістю від користувача
- Додавання даних через файл та Web-формою

Висновок до розділу

У розділі описано ознайомчу інформацію по дипломному проекту. Його цілі та задачі розробки з переліком, процес діяльності, функціональну модель. Був зроблений огляд існуючих аналогів. Показані спільні та відмінні риси.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

Вхідні дані представлені у таблиці 2.1.

Таблиця 2.1 – вхідних даних.

Назва даних	Опис
Дані авторизації	Отримуються автоматично з сторонній сервіс Google або Facebook
Дані про товари	Додаються завантаженням файлу формату Excel (.xls) з рядками Назви та ціни продукту.
Дані про товари	Додаються або змінюються через Web-форму.
Коментарі	Коментарі про магазин додаються авторизованими користувачами через Web-форму

2.2 Вихідні дані

Вихідні дані представлені в таблиці 2.2.

Таблиця 2.2 – вихідні дані

Назва даних	Опис
Пошукові результати	Пошукові результати по запиту
Інформаційні повідомлення про статус перевірки	Відправляються модераторами власникам закладів

Продовження Таблиці 2.2

Відгуки до магазину	Відображаються при пошуку продукту біля магазину

2.3 Опис структури бази даних

Для зберігання даних у проекті використовується реляційна база даних з такими таблицями Product, Product_type, Company, Store, Users, Contacts, Comments.

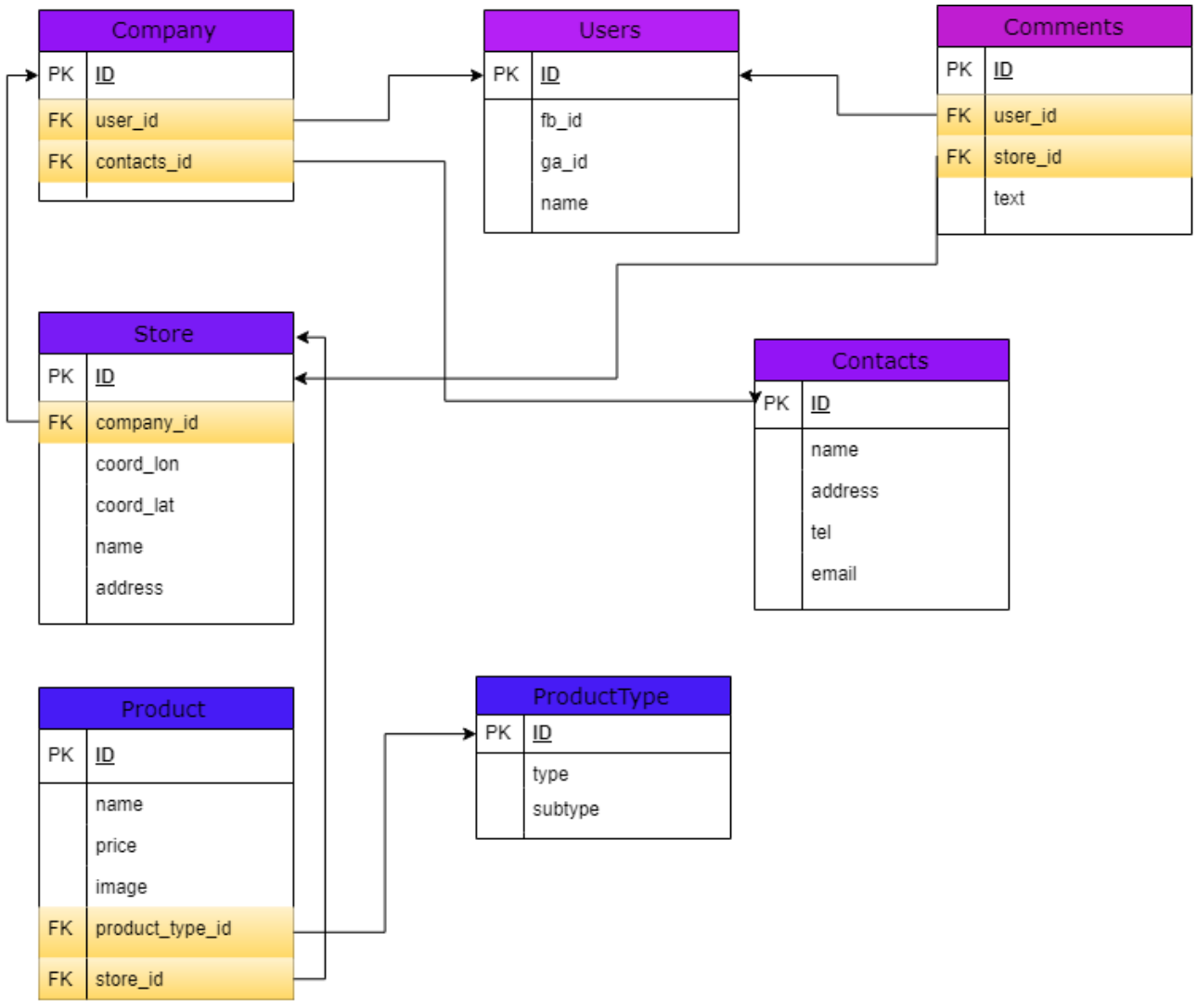

Рисунок 2.1 – ER-діаграма бази даних

Таблиця **Product** має поля:

Id – первинний ключ, унікальне значення, INTEGER

name – назва продукту, не пусте значення до 255 символів VARCHAR(255)

image – посилання на зображення, VARCHAR(255)

store_id – зовнішній ключ, не пусте значення, INTEGER

product_type_id - зовнішній ключ, не пусте значення, INTEGER

Таблиця **Product_type** має поля:

Id – первинний ключ, унікальне значення, INTEGER

type – тип продукту, не пусте значення до 100 символів VARCHAR(100)

subtype – підтип продукту, не пусте значення до 100 символів VARCHAR(100)

Таблиця **Store** має поля:

Id – первинний ключ, унікальне значення, INTEGER

company_id - зовнішній ключ, не пусте значення, INTEGER

coord_lon – довгота, не пусте значення, DOUBLE PRECISION

coord_lat – широта, не пусте значення, DOUBLE PRECISION

name – назва, не пусте значення до 255 символів VARCHAR(255)

address – адреса, не пусте значення до 255 символів VARCHAR(255)

Таблиця **Company** має поля:

Id – первинний ключ, унікальне значення, INTEGER

user_id - зовнішній ключ, не пусте значення, INTEGER

contacts_id - зовнішній ключ, не пусте значення, INTEGER

Таблиця **Users** має поля:

Id – первинний ключ, унікальне значення, INTEGER

fb_id – значення до 512 символів VARCHAR(512)

ga_id – значення до 512 символів VARCHAR(512)

name – значення до 255 символів VARCHAR(255)

Таблиця **Comments** має поля:

Id – первинний ключ, унікальне значення, INTEGER

user_id – зовнішній ключ, не пусте значення, INTEGER

store_id – зовнішній ключ, не пусте значення, INTEGER

text – текст відгуку, значення до 255 символів VARCHAR(255)

Таблиця **Contacts** має поля:

Id – первинний ключ, унікальне значення, INTEGER

name – назва компанії, значення до 255 символів VARCHAR(255)

address – адреса компанії, значення до 255 символів VARCHAR(255)

tel – телефон, значення до 30 символів VARCHAR(30)

email – ел. пошта, значення до 200 символів VARCHAR(200)

Висновок до розділу

У розділі були описані вхідні та вихідні дані. Описана структура бази даних та пояснення до кожної таблиці.

МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

Для реалізації програмного забеспечення були виділені наступні задачі:

– Пошук товару за назвою, зважаючи на те, що введені дані можуть незначно відрізнятись від назви того ж товару у базі даних.

(Приклад: «БАЛИК ДАРНИЦЬКИЙ» — «БАЛИК ДАРНИЦЯ»)

– Розрахування відстаней від місця знаходження користувача до магазинів по GPS-координатам.

3.2 Математична постановка задачі

3.2.1 Пошук товару за назвою з незначною різницею

Шукаючи конкретний продукт, дані для пошуку, введені користувачем можуть незначно відрізнятись від назви ого ж товару у базі даних. Наприклад у базі існує продукт під назвою «БАЛИК ДАРНИЦЬКИЙ», проте користувач може ввести «БАЛИК ДАРНИЦЯ». Результат повинен бути однаковий.

3.2.2 Розрахунок відстаней за координатами

При знаходженні точок продажу з потрібним користувачу товаром потрібно знайти відстані від поточного місцезнаходження користувача до магазинів, які задовольняють критеріям пошуку.

3.3 Обґрунтування методу розв’язання

Для вирішення задачі 3.2.1 було знайдено декілька алгоритмів нечіткого пошуку у тексті таких як Відстань Левенштейна[1], алгоритм Касаі[28] з будуванням суфіксного дерева, але всі вони вимагають значного використання ресурсів, оскілки пошук потрібно проводити по всьому списку продуктів. Тому для розв’язання цієї був використаний значно простіший метод пошуку префіксу з обмеженням.

Для розв’язання задачі 3.2.2 були знайдені методи Вінсенті[2] та формула гаверсинуса[3]. Для вирішення задачі була використана саме формула гаверсинуса, оскільки для результату пошуку потрібна не точність до міліметрів, а швидкість виконная операції.

3.4 Роз’язання математичних задач

3.4.1 Пошук товару за назвою з незначною різницею

Для пошуку був використаний такий алгоритм:

- Починається пошук повної відповідності. Успіх – завершити виконання, перейти до наступного кроку якщо ні.
- Порахувати кількість символів у запиті. Якщо поточна кількість більше 2/3 – перейти до наступного кроку. Інакше – завершити виконання.
- Видалити останній символ. Виконати пошук за шаблоном. Якщо нічого не знайдено – повернутись до кроку 2, інакше – завершити виконання.

3.4.2 Розрахунок відстаней за координатами

Для розрахунку відстані використовується формула гаверсинусів.

$$\operatorname{hav}\left(\frac{d}{r}\right) = \operatorname{hav}(\varphi_2 - \varphi_1)$$

(3.1)

Де d - це центральний кут між двома точками, що лежать на великому колі, r - радіус сфери, φ_1 і φ_2 - широта першої і другої точок в радіанах, λ_1 і λ_2 - довгота першої і другої точок в радіанах.

Знайдемо d:

$$d = 2r \arcsin\left(\sqrt{\operatorname{hav}(\varphi_2 -}\right.$$

(3.2)

Замінемо гаверсинус за визначенням:

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 -}{2}\right.}\right.$$

(3.3)

Висновок до розділу

У розділі математичного забеспечення були оглянуті математичні задачі дипломного проекту. Оглянуті різні варіанти вирішення поставлених задач. Обгрутовано вибір поточних рішень.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

Розробку програмного забеспечення можна логічно розділити на дві частини: клієнтську та серверну.

Для розробки клієнтської частини використовувались такі засоби:

- **HTML5** діалект гіпертекстової розмітки для будування структури веб-сторінок і додатків. HTML5 наслідує можливості ранніх версій HTML з додаванням оновненого API. [7]
- **SCSS** дає можливість використовувати змінні, вкладені правила, міксини, інлайновие обсяги імпорту і багато іншого, все з повністю сумісним з CSS синтаксисом. [8]
- **JavaScript** — прототипна, об'єктно-орієнтована, динамічна мова програмування. Використовується для опису поведінки веб-елементів що взаємодіють з користувачем, надаючи можливість змінювати поведінку сторінки та взаємодіяти с користувачем. [9]

- **ReactJS —** бібліотека, написана на мові JavaScript яка дає можливість розробляти якісні та стабільні компоненти взаємодії з користувачем, які можуть оновлювати сторінку частково.[27]

- **NodeJS** — написане на JavaScript та побудоване на V8 середовище виконання, яке допомогає запускати застосунки JavaScript та допомогає вирішувати основні веб-задачі. [10]

- **ESlint** — це інструмент для виявлення і звітування по шаблонам, знайденим в коді ECMAScript / JavaScript, з метою зробити код більш послідовним і уникнути помилок. [11]
- **Yarn** — кешує кожен пакет, який він завантажує, тому ніколи не потрібно його знову завантажувати. Він також паралелізує операції для максимального використання ресурсів. [12]
- **webpack** — це статичний модуль для програм JavaScript. Коли webpack обробляє програму, він внутрішньо створює граф залежностей, який відображає кожен модуль, необхідний вашому проекту, і генерує один або більше пакетів. [13]
- **Jest** — фреймфорк для тестування JavaScript. [14]
- **Puppeteer** — це бібліотека NodeJS, яка надає високорівневий API для керування Chrome або Chromium над протоколом DevTools. [15]
- **HuskyJS** — дозволяє легко створювати Git hooks. Git hooks- це скрипти, які виконуються до або після події. [16]
- **LernaJS** — інструмент для керування проектами JavaScript з багатьма пакетами. [17]

Для розробки серверної частини використовувались такі засоби:

- **Python** — Одна з найвідоміших інтерпритованих мов програмування з динамічною типізацією з наголосом на простоту та наглядність. Має велику кількість якісних бібліотек. [18]
- **Nginx —** вільний веб-сервер і проксі-сервер. [6]
- **Gunicorn** — WSGI HTTP сервер для мови Python. [5]

- **Flask** — легкий фреймворк для розробки веб-додатків, написаний на мові Python. [19]

- **Supervisor —** це система клієнт / сервер, яка дозволяє користувачам контролювати і керувати деякими процесами на UNIX-подібних операційних системах. [4]
- **Psycopg2** — адаптер бази даних PostgreSQL для мови програмування Python. [20]
- **Jinja2** — мова та інстумент використання шаблонів. Стандарно використовується у Flask. [21]
- **argon2-cffi** — захищений хеш паролів для Python. [22]
- **Flask-Dance** — бібліотека авторизації через популярні сервіси, такі як Google Account, Facebook та інші. [23]
- **Flask-SQLAlchemy** — інструментарій SQL та об'єктно-реляційне відображення(ORM) для мови програмування Python. [24]
- **requests** — дозволяє надсилати запити без необхідності ручної праці. [25]
- **numpy** — Всесвітньовідома Python бібліотека для роботи з великими об’ємами даних та важкими математичними операціями. [26]

4.2 Вимоги до технічного забезпечення

Вимоги до серверу:

Для роботи системи необхідні наступні параметри системи:

– Процесор с частотою 1,8 ГГц або більш потужний.

– ОЗУ об’ємом 1 ГБ (2,5 ГБ для роботи на віртуальній машині)

– 20 Гб доступного простору на жорстким диску

– Жорсткий диск с частотою обертання 5 400 об/хв або більше

– Відеокарта с підтримкою DirectX 9 и розширення екрану 1024x768 або вище

Також необхідне встановлене наступне програмне забеспечення:

- Ubuntu 18.04 або аналогічна
- Python 3.7
- Nginx 1.17.0
- Flask 1.0.2
- gunicorn 19.9.0
- Supervisor 4.0.3
- PostgreSQL 10.6

  Та усі вище названі Python пакети актуальної версії.

Вимоги до клієнту:

Наявність одного з веб-браузерів:

- Safari 12.0+;
- Firefox 66.0+;
- Google Chrome 74.0+;
- Chromium 71+;

4.3 Архітектура програмного забезпечення

4.3.1 Діаграма розгортання

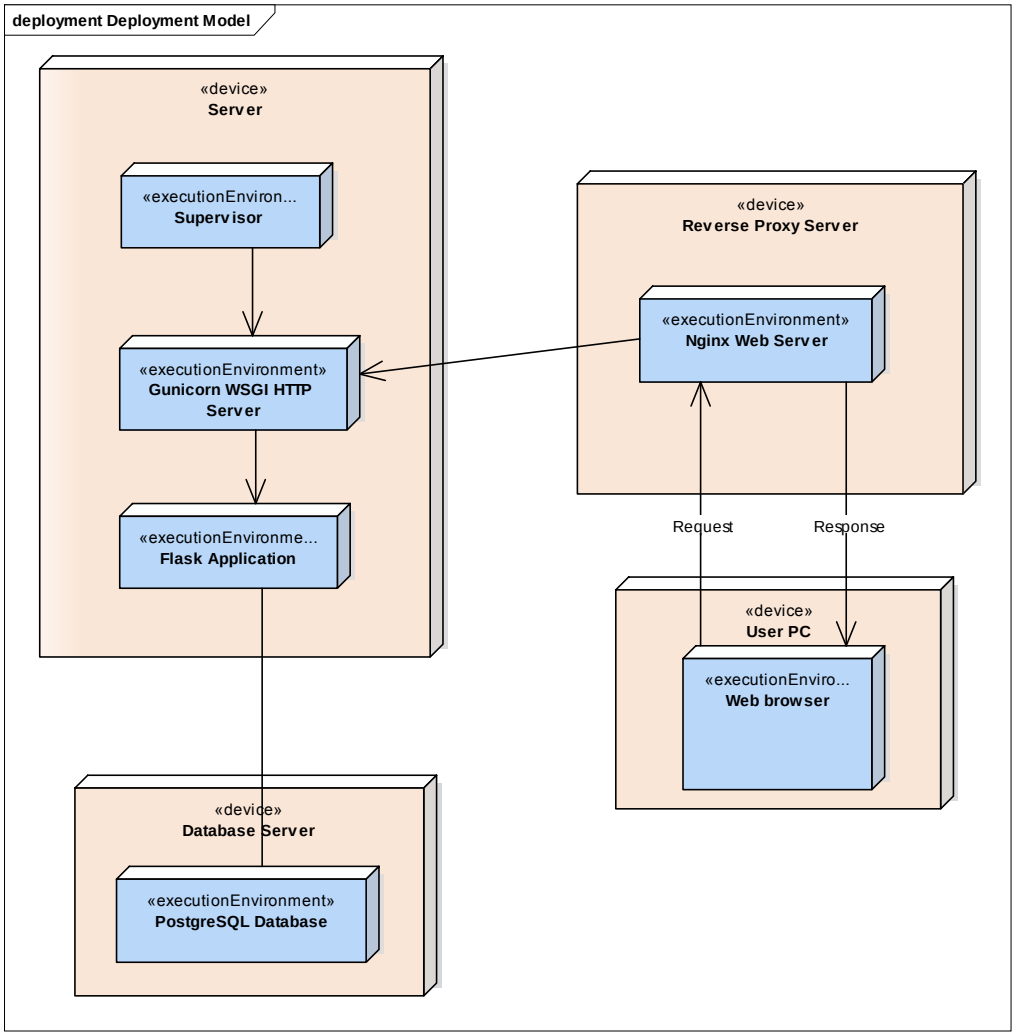

Рисунок 4.3.1 – Діаграма розгортання

На діаграмі розгортання зображені ключові компоненти сервера:

- Supervisor – призначений для моніторингу та управління Веб-сервера gunicorn
- Gunicorn WSGI HTTP Server – розгортає Flask додаток
- Flask Application – серверна частина проекту
- PostgreSQL – реляційна база даних
- Nginx Web Server – виконує роль зворотнього проксі-сервера

4.3.2 Діаграма послідовності

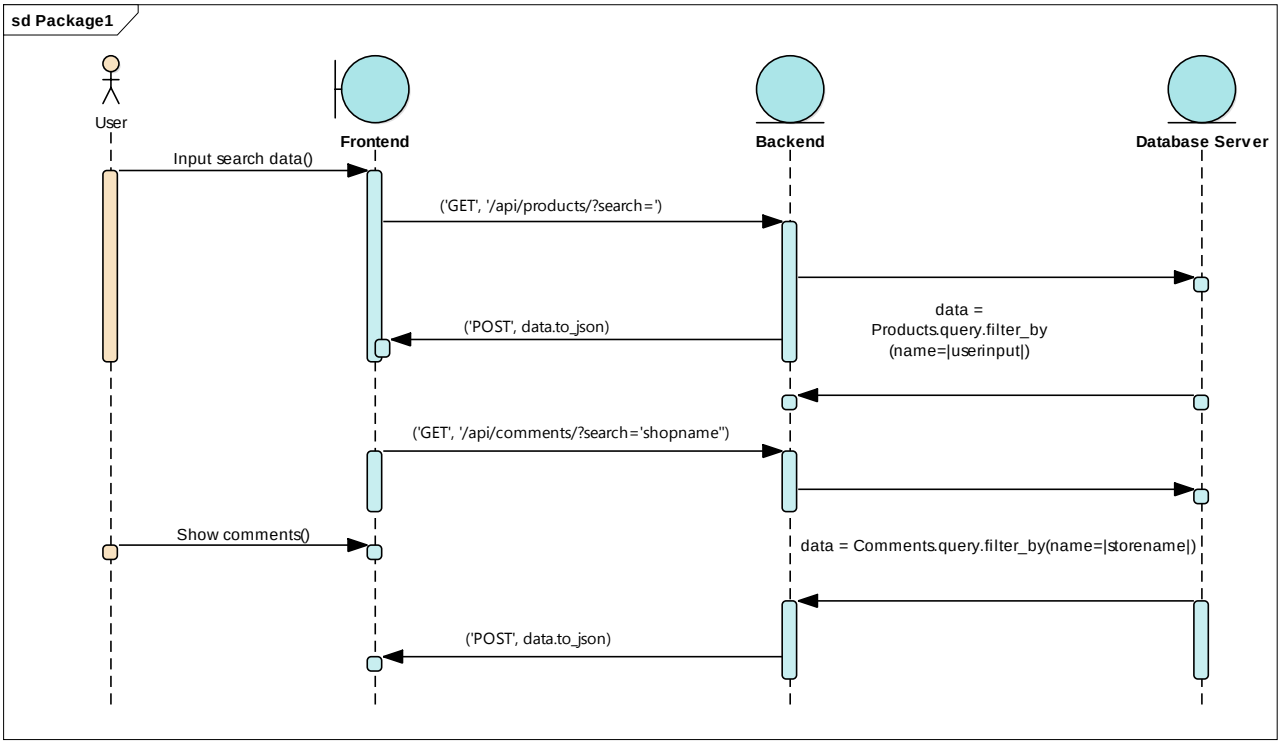

Рисунок 4.3.2 – Діаграма послідовності

На діаграмі послідовності зображене послідовне виконання пошуку товару та перегляд коментарів до вибраного магазину. Спочатку користувач вводить дані для пошуку та нажимає за кнопку пошуку. Клієнтська частина передає HTTP GET запит на знаходження усіх продуктів з параметром ім’я продукту. Серверна частина зчитує таблицю Products і формує результат у об’єкт классу Product. Далі цей об’єкт конвертується та передається HTTP POST запитом до клієнтської частини у вигляді JSON. Результати відображаються користувачу.

Потім користувач вибирає один с вибраних магазинів та натискає кнопку перегляду відгуків. Клієнтська частина передає HTTP GET запит на знаходження усіх коментарів з параметром ім’я магазину. Серверна частина

зчитує таблицю Comments і формує результат у об’єкт классу Comment. Далі цей об’єкт конвертується та передається HTTP POST запитом до клієнтської частини у вигляді JSON. Результати відображаються користувачу.

4.3.3 Діаграма варіантів використання

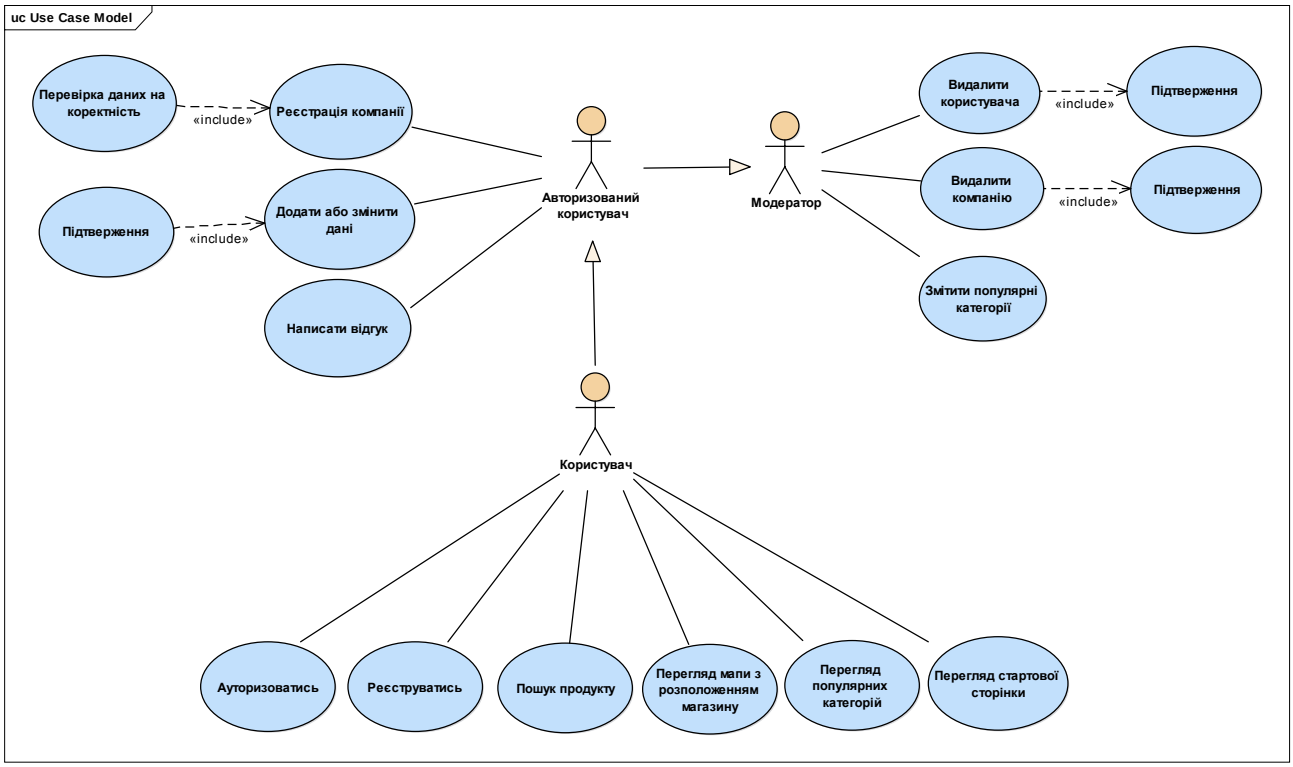

Рисунок 4.3.3 – Діаграма варіантів використання

У системі є три ролі: незареєстрований користувач, авторизований користувач та модератор.

У незареєстрованого користувача є доступ тільки до основної функціональності системи – реєстрації\ауторизації, перегляду популярних категорії та пошуку.

У авторизованого користувача крім основних є значно більше можливостей. Можливість залишати відгуки до магазинів, можливість зареєструвати свою компанію та змінювати її вміст даних.

Модератор має доступ до усього функціоналу системи, який включає в себе сторінку модерації з можливостями змінювати та видаляти асортимент магазинів, самі магазини, компанії, користувачів та їх відгуки.

Висновок до розділу

У розділі були описані засоби розробки клієнтської та серверної частини, вимоги до програмного і технічного забезпечення, а також, для кращого розуміння архітектури програмного забезпечення були наведені діаграми з поясненнями. А саме: діаграма розгортання, діаграма послідовності та діаграма варіантів використання.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

На головній сторінці знаходяться форми авторизації\реєстрації користувача, строка пошуку, популярні категорії продуктів. Користувач може здійснити пошук одразу, або авторизуватисьРисунок 5.1 – Стартова сторінка

Авторизація та реєстрація виконуються за допомогою облікових записів Google Account або соціальної мережі Facebook. Для цього потрібно заздалегідь авторизуватись у одній з цих мереж.

Рисунок 5.2 – Сторінка авторизації

Після завершення простої реєстрації або авторизації користувач перенаправляється на стартову сторінку вже як авторизований користувач.Рисунок 5.3 – Вигляд стартової сторінки авторизованого користувача

Усі зареєстровані користувачі, крім функції пошуку, мають можливість реєструвати у системі свої компанії та магазини. Для цього потрібно заповнити просту форму з ПІБ власника, назвою компанії та сканами документів, підтверджуючих власність компанією. Після успішної перевірки компанія буде додана у базу даних системи.

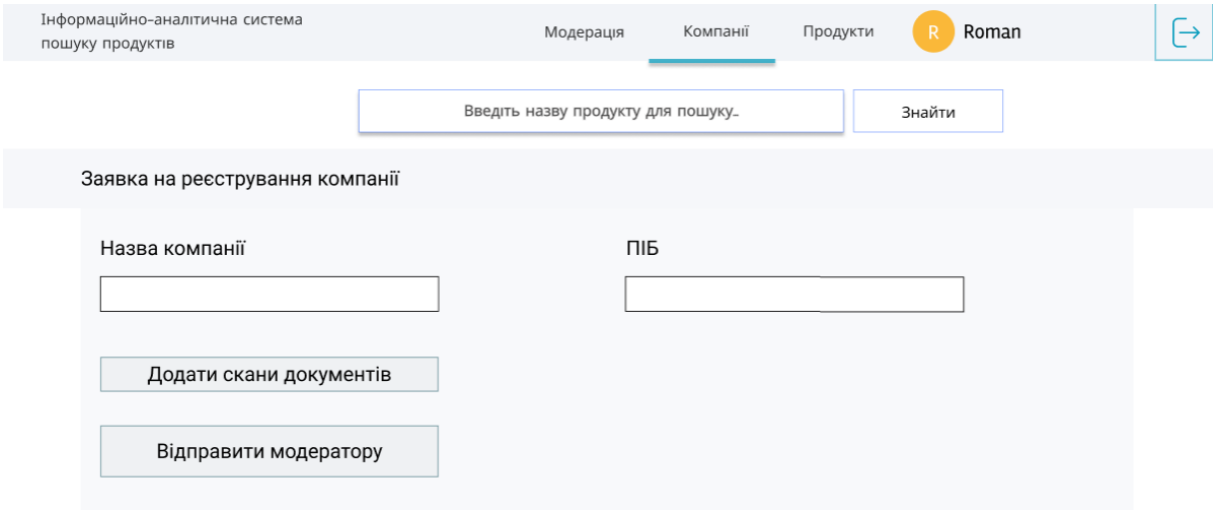

Рисунок 5.4 – Заявка на реєстрування компанії

Після успішної перевірки компанія буде додана у базу даних системи.

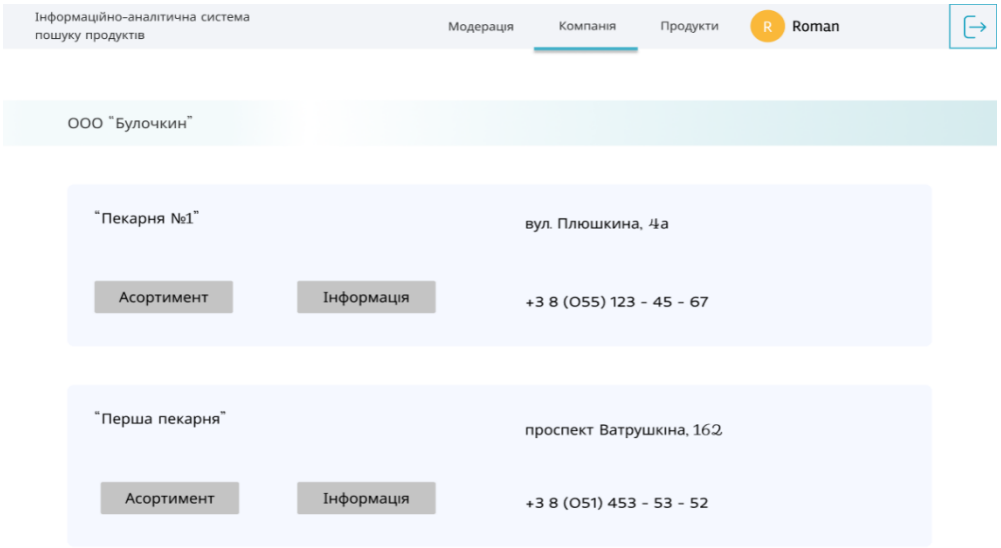

Рисунок 5.5 – Сторінка созданої компанії

Власник зареєстрованої компанії має можливість переглянути, додати або видалити продукти з свого магазину.

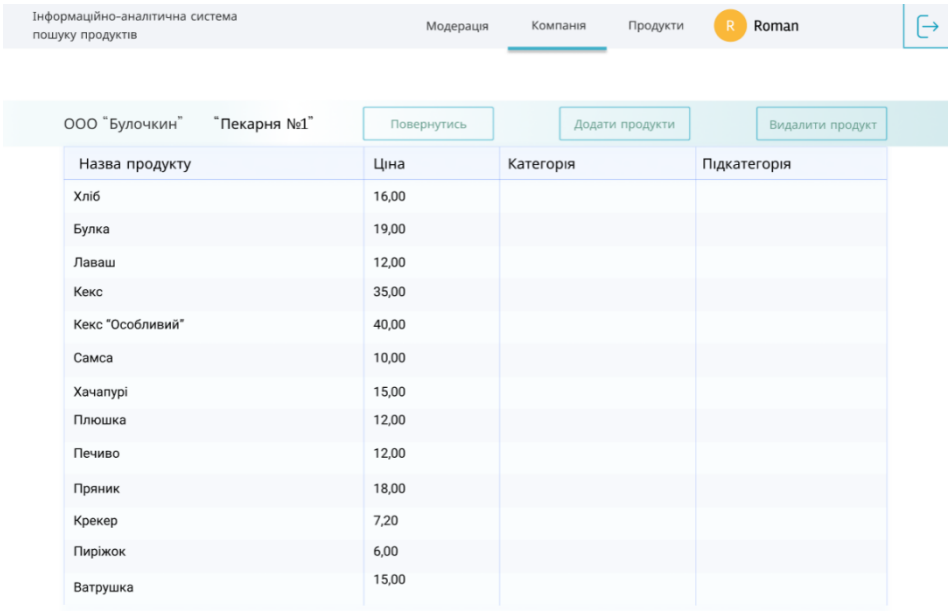

Рисунок 5.6 – Сторінка магазину компанії

Також доступ до перегляду та зміни усіх даних має модератор. Крім основних задач, модератори сортують несортовані продукти на категорії та підкатегорії.

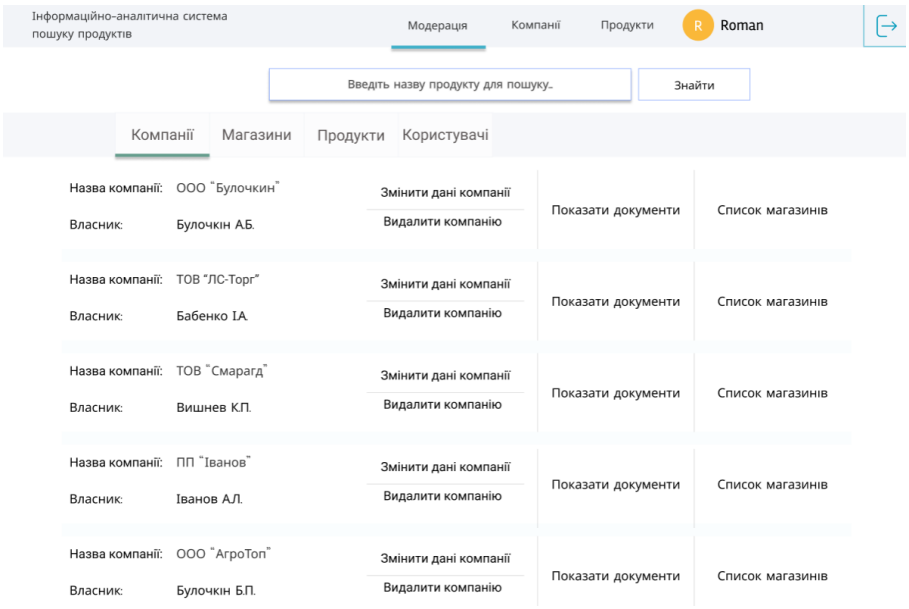

Рисунок 5.6 – Сторінка перегляду компаній модератора

Оскільки можливість «Видалити компанію» є досить відповідальною – при спробі видалити компанію або магазин з’являється спливаюче вікно з проханням ввести повну назву видаляємого об’єкта.

При натисканні «Список магазинів» модератора перемістить на наступну сторінку зі списком усіх магазинів вибраної компанії.

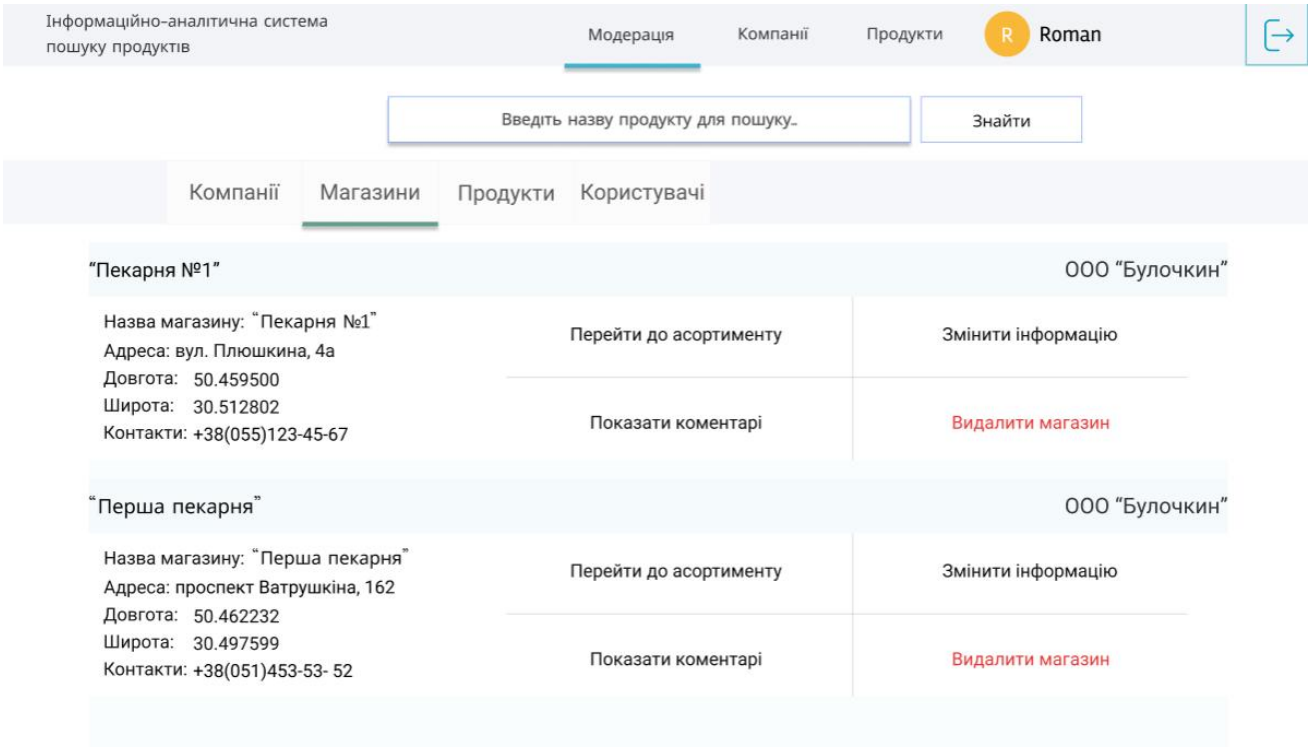

Рисунок 5.7 – Сторінка перегляду магазинів компанії модератора

На вкладці виведений список усіх магазинів компанії та головні інструменти модератора. При натисканні на «Перейти до асортименту» та «Показати коментарі» відкриються відповідні сторінки по цьому магазину.

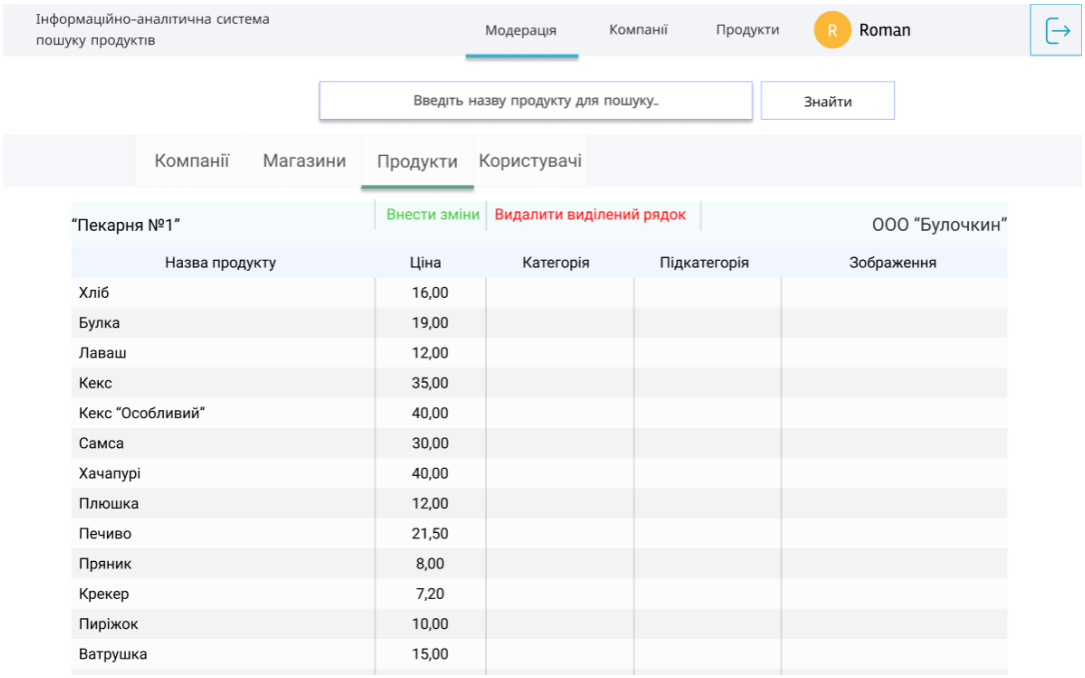

Рисунок 5.8 – Сторінка перегляду продуктів магазину для модератора

Модератор має можливість власноруч змінювати будь-які значення таблиці та зберігати їх за допомогою кнопки «Внести зміни». Також є можливість видаляти виділений рядок фокусом на потрібний рядок і натиском кнопки «Видалити виділений рядок».

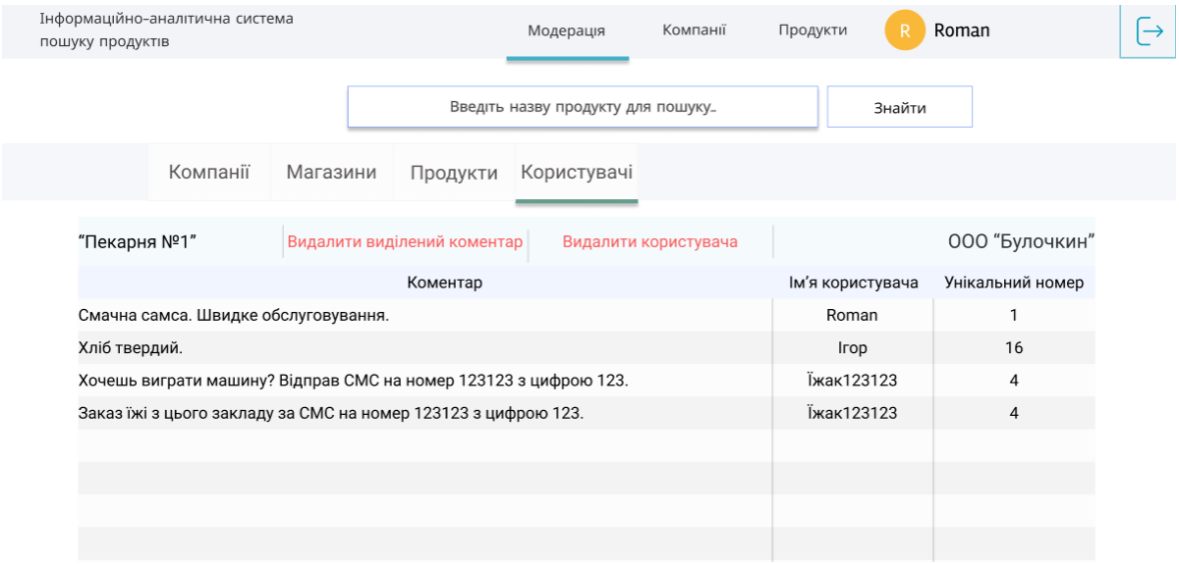

Рисунок 5.9 – Сторінка перегляду відгуків магазину для модератора

По аналогією з попередньою формою тут є кнопки видалення відгуку та користувача. Сам пошук продукту в системі досить простий та інтуїтивно зрозумілий. Для запуску потрібно ввести у відповідну строку назву продукту та натиснути «Знайти».

Рисунок 5.10 – Початок пошуку продукту

Рисунок 5.11 – Результати пошуку продукту

Після пошуку відкриється форма з результатами. Під кожним магазином є кнопки «Карта», яка відкриває карту з місцерозташуванням даного магазина та «Відгуки», яка відкриває вікно відгуків.

Рисунок 5.12 – Карта

Рисунок 5.13 – Відгуки

5.2 Випробування програмного продукту

5.2.1 Мета випробувань

Метою випробувань є перевірка відповідності функцій інформаційно-аналітичної системи пошуку продуктів вимогам технічного завдання.

5.2.2 Загальні положення

Випробування проводяться на основі наступних документів:

- ГОСТ 34.603-92. Інформаційна технологія. Види випробувань автоматизованих систем;
- ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3 Результати випробувань

Під час тестування була перевірена функціональна складова системи. Нижче наведені результати випробувань у вигляді таблиць.

Таблиця 5.1 – Реєстрація в системі

Мета тесту	**Перевірка реєстрації в системі**
Початковий стан моделі	Початкова сторінка з формою реєстрації
Вхідні дані:	Ім’я користувача, авторизований Facebook аккаунт
Схема проведення тесту:	Натиснути «Увійти через Facebook», підтвердити, ввести ім’я.

Продовження таблиці 5.1

Очікуваний результат:	Повідомлення про успішну реєстрацію
Стан моделі після проведення випробувань:	Повідомлення про успішну реєстрацію, закриття форми авторизації.

Таблиця 5.2 – Коректність пошуку

Мета тесту	Перевірка коректності пошуку
Початковий стан моделі	Відкрита початкова сторінка, ауторизований користувач.
Вхідні дані:	Пошуковий запит «Кекс», геодані
Схема проведення тесту:	Ввести у поле вводу форми пошуку. Натиснути кнопку «Знайти»
Очікуваний результат:	Вивід результатів пошуку.
Стан моделі після проведення випробувань:	Вивід трьох результатів пошуку.

Таблиця 5.3 – Робота карт

Мета тесту	Перевірка роботи карт
Початковий стан моделі	Відкритий результат пошуку.
Вхідні дані:	Відсутні.
Схема проведення тесту:	Обрати один з результатів та натиснути кнопку «Карта»
Очікуваний результат:	Спливаюче вікно з відміткою магазину на карті.
Стан моделі після проведення випробувань:	Спливаюче вікно з відміткою магазину на карті.

Таблиця 5.4 – Функція «Повернутись»

Мета тесту	Перевірка функції «Повернутись»
Початковий стан моделі	Відкритий результат пошуку.
Вхідні дані:	Відсутні.
Схема проведення тесту:	Натиснути «Повернутись»
Очікуваний результат:	Початкова сторінка

Стан моделі після проведення випробувань:	Початкова сторінка

Таблиця 5.5 – Модерація

Мета тесту	**Перевірка функції «Модератор»**
Початковий стан моделі	Відкрита початкова сторінка, ауторизований користувач.
Вхідні дані:	Відсутні.
Схема проведення тесту:	Натиснути «Модерація»
Очікуваний результат:	Сторінка модератора
Стан моделі після проведення випробувань:	Сторінка модератора

Висновок до розділу

У розділі було розглянуто інструкцію користувача, ознайомлення з можливостями системи.

Було успішно виконане тестування головних функцій системи.

ЗАГАЛЬНІ ВИСНОВКИ

У сучасній Інтернет-мережі на сьогоднішній день представлений широкий спектр сайтів, що автоматизують систему пошуку замовлень і просування товару . Ці сайти різняться за рівнем складності структури і технологій створення.

Тема дипломної роботи була обрана у зв’язку з актуальністю та потребою розробки пошукового системи найближчих за знаходженням потрібних споживчих продуктів . Метою даної роботи стала розробка пошукового сайту для вимогливих споживачів, обмежених в часі на пошук потрібного бажаного продукту.

Для досягнення мети були поставлені та виконані таки завдання:

– огляд сайтів схожої тематики;

– виконання порівняльного аналізу програмних засобів побудови сайтів;

– проектування структури і сервісів сайту;

– реалізація та впровадження сайту;

Результатами роботи став розроблений і впроваджений сайт для пошуку конкретного споживчого продукту, який відповідає всім вимогам замовника та знаходиться на найближчий відстані від замовника за його геолокацією.

ПЕРЕЛІК ПОСИЛАНЬ

1. Пошукова система [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Пошукова_система

2. Для чого нам потрібні пошукові системи? [Електронний ресурс] – Режим доступу до ресурсу: https://sites.google.com/site/posukovisistemi1/

3. Інформаційно-аналітична система [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Інформаційно-аналітична_система

4. supervisor [Електронний ресурс] – Режим доступу до ресурсу: http://supervisord.org/

5. gunicorn [Електронний ресурс] – Режим доступу до ресурсу: https://docs.gunicorn.org/en/stable/index.html

6. Nginx [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Nginx

7. HTML [Електронний ресурс] – Режим доступу до ресурсу: https://html.spec.whatwg.org/

8. Sass [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Sass

9. JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/JavaScript

10. Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/nodejs/node

11. eslint [Електронний ресурс] – Режим доступу до ресурсу: https://eslint.org/docs/user-guide/getting-started

12. Yarn [Електронний ресурс] – Режим доступу до ресурсу: https://yarnpkg.com/en/docs/getting-started

13. webpack [Електронний ресурс] – Режим доступу до ресурсу: https://webpack.js.org/concepts

14. jest [Електронний ресурс] – Режим доступу до ресурсу:

https://jestjs.io/

15. puppeteer [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/GoogleChrome/puppeteer

16. husky [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/typicode/husky

17 lerna [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/lerna/lerna#about

18 Python [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org/

19 Flask [Електронний ресурс] – Режим доступу до ресурсу: http://flask.pocoo.org/

20 psycopg2 [Електронний ресурс] – Режим доступу до ресурсу: https://pypi.org/project/psycopg2/

21 Jinja [Електронний ресурс] – Режим доступу до ресурсу: http://jinja.pocoo.org/docs/2.10/

22 argon2-cffi [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/hynek/argon2-cffi

23 https://flask-dance [Електронний ресурс] – Режим доступу до ресурсу: https://flask-dance.readthedocs.io/en/latest/

24. SQLAlchemy [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/SQLAlchemy

25. requests [Електронний ресурс] – Режим доступу до ресурсу: https://2.python-requests.org/en/master/

26. NumPy [Електронний ресурс] – Режим доступу до ресурсу: https://www.numpy.org/

27. React [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/facebook/react/

Додаток А

Тексти програмного коду

Інформаційно – аналітична система пошуку продуктів

(Найменування програми (документа))

DVD-R

(Вид носія даних)

33 аркуша, 3.5 мб

(Обсяг програми (документа) 33, арк., 2780 Кб)

Київ – 2019 року

Лістинг програми app.py

```
from flask import Flask, render_template, request,
redirect, session, url_for, jsonify

import psycopg2
```

```
from urllib.parse import unquote, urlparse

from os import environ

import api


url = urlparse(environ.get('DATABASE_URL'))

db = "dbname=%s user=%s password=%s host=%s " %
(url.path[1:], url.username, url.password, url.hostname)

# db = "host='localhost' dbname='test_db'
user='test_user' password='qwerty'"

conn = psycopg2.connect(db)

cur = conn.cursor()


app = Flask(__name__, static_url_path='/static',
static_folder='templates/static')

app.secret_key = 'shazam'




# landing page

@app.route(r'/')

def landing_page():

    return render_template('index.html')
```

```
# login page
@app.route(r'/login', methods=['GET', 'POST'])
def login():
    error = None
    if request.method == 'POST':
        # logs
        un = request.form['username']
        pwd = request.form['password']
        print(un)
        print(pwd)
        # check if login exists
        check_username = f"select exists(select login
from users where login='{un}' limit 1)"
        # return users.id if login\pwd correct, else
TypeError
        check_login = f"select id from users where
(login='{un}' and password='{pwd}')"
        # logs
        print(f'check_username={check_username}')
        print(f'check_login={check_login}')
        # execute q
```

```
        cur.execute(check_login)
      res = cur.fetchone()
        try:
            if res[0]:
                print(res[0])
                print('Login successful')
                return redirect(url_for('welcome'))
        except TypeError:
            error = 'Login\Password combination not
exists.'
            print(error)

    elif request.method == 'GET':
        pass
    pass
    return render_template('login.html', error=error)


# new user registration
@app.route(r'/registration')
def registration():
    pass
    # return render_template('registration.html')
```

```
# admin page
@app.route(r'/admin')
def admin_menu():
    pass
    return render_template('admin.html')



# successful login redirect page
@app.route(r'/welcome')
def welcome():
    return render_template('welcome.html')



# api
@app.route('/api/<string:apifunc>/')
def do(apifunc):
    query = tuple(request.args.values())
    getattr(api, apifunc)(request.method, query,
request.content_type, request.headers)
    return redirect(url_for('landing_page'))
```

```
if __name__ == '__app__':
    app.run(debug=True)
```

Лістинг програми api.py

```
from flask import Flask, render_template, request,
redirect, session, url_for, jsonify
import psycopg2
from urllib.parse import unquote, urlparse
from os import environ
import api

url = urlparse(environ.get('DATABASE_URL'))
db = "dbname=%s user=%s password=%s host=%s " %
(url.path[1:], url.username, url.password, url.hostname)
# db = "host='localhost' dbname='test_db'
user='test_user' password='qwerty'"
conn = psycopg2.connect(db)
cur = conn.cursor()

app = Flask(__name__, static_url_path='/static',
static_folder='templates/static')
app.secret_key = 'shazam'
```

```
# landing page
@app.route(r'/')
def landing_page():
    return render_template('index.html')
# login page
@app.route(r'/login', methods=['GET', 'POST'])
def login():
    error = None
    if request.method == 'POST':
        # logs
        un = request.form['username']
        pwd = request.form['password']
        print(un)
        print(pwd)
        # check if login exists
        check_username = f"select exists(select login
from users where login='{un}' limit 1)"
        # return users.id if login\pwd correct, else
TypeError
        check_login = f"select id from users where
(login='{un}' and password='{pwd}')"
```

```
        # logs
        print(f'check_username={check_username}')
        print(f'check_login={check_login}')
        # execute q
      cur.execute(check_login)
        res = cur.fetchone()
        try:
            if res[0]:
                print(res[0])
                print('Login successful')
                return redirect(url_for('welcome'))
        except TypeError:
            error = 'Login\Password combination not
exists.'
            print(error)

    elif request.method == 'GET':
        pass
    pass
    return render_template('login.html', error=error)
```

```
# new user registration
@app.route(r'/registration')
def registration():
    pass
  # return render_template('registration.html')
# admin page
@app.route(r'/admin')
def admin_menu():
    pass
    return render_template('admin.html')



# successful login redirect page
@app.route(r'/welcome')
def welcome():
    return render_template('welcome.html')



# api
@app.route('/api/<string:apifunc>/')
def do(apifunc):
```

```
    query = tuple(request.args.values())
    getattr(api, apifunc)(request.method, query,
request.content_type, request.headers)
    return redirect(url_for('landing_page'))
```

Лістинг програми etc.py

```
import json


class BaseData:
    def __init__(self, current_page, total_pages, limit):
        """Constructor"""
        self.current_page = current_page
        self.total_pages = total_pages
        self.limit = limit
        self.results = []

    def to_json(self):
        return json.dumps(self.__dict__,
ensure_ascii=False)



class Products(BaseData):
```

```
    """docstring"""

    def __init__(self, current_page, total_pages, limit):
        super().__init__(current_page, total_pages,
limit)

    # value - (db row) tuple of values, args - ordered
tuple of names
    def add_to_results(self, value: tuple, args: tuple):
        self.results.append({i[1]: value[i[0]] for i in
enumerate(args, 0)})

    # return dict-like string for post query values
    def prepare_post_query(self):
        return str(self.results[0])
```

Лістинг програми sql_query.py

```
def select_products_by_name(name, limit=1000, page=1) ->
str:
    """
    Select rows with pagination where name match.
    Pagination disabled by default.
```

```
    :param name: the name of product to be searched
    :param limit: search result limit for page, 1000 by
default
    :param page: number of result page, should be used
with limit param, 1 by default
    :return: sql query string : product_name,
product_price and store_name.
    """
    temp = "SELECT foo.name, foo.price, store.name "
    temp += "FROM (SELECT * FROM  product JOIN "
    temp += f"(SELECT id FROM product WHERE name =
'{name}' "
    temp += f"ORDER BY id LIMIT {int(limit)} OFFSET
{int(limit) * (int(page) - 1)}) as b "
    temp += "ON b.id = product.id) as foo "
    temp += "INNER JOIN store "
    temp += "ON foo.store_id = store.id;"

    return temp


def check_username_exists(username: str) -> str:
```

```
    """
    Check if username exists in users table. Return
boolean value.

    :param username: user login
    :return: sql query string : boolean
    """
    temp = "SELECT EXISTS ("
    temp += "SELECT login FROM users WHERE "
    temp += f"login = '{username}');"
    return temp


def check_login_password_combination(username: str,
password: str) -> str:
    """
    Check login\password combination.

    :param username: user login
    :param password: user password
    :return: sql query string : user_id.
    """
    temp = "SELECT id FROM users WHERE "
```

```
    temp += f"(login = '{username}' AND password =
'{password}')"

    return temp


def products_found(name: str) -> str:
    """
    Count of product found

    :param name: product name
    :return: sql query string: count integer
    """
   temp = f"SELECT COUNT(*) FROM product WHERE name =
'{name}';"

    return temp
def distinct_products(limit=20):
    """
    Select by unique product name
```

```
    :param limit: number of products selected, 20 by
default
    :return: sql query string: list of product name,
product price
    """
    temp = f"SELECT DISTINCT name, price from product
limit {int(limit)}"

    return temp

def insert_row(table: str, header: dict) -> str:
    """
    Insert values into table

    :param table: name of table
    :param header: application/json header
    :return: sql query string

 """
    print(header)
    # value: tuple with values without PK_id
    value = tuple([i for i in header.values()])
```

```
    # col_name: tuple with column names
    col_name = tuple(i for i in header.keys())
    print(value)
    print(col_name)
    temp = f"INSERT INTO {table} {col_name}
".replace("'", "")
    temp += f"VALUES {value}"
    print(temp)
    return temp



# TODO check id exist
def delete_row(table: str, element_id: int):
    temp = f"DELETE FROM {table} WHERE id = {element_id}"
    return temp
```

Лістинг програми app.js

```
import React, { Component } from 'react';
import './App.css';
import Main from './components/Main';
import { connect } from 'react-redux';
```

```
import { HashRouter as Router, Route } from 'react-
router-dom';
import Product from './components/Product';
import Company from './components/Company';
import Dashboard from './components/Dashboard';
import Admin from './components/Admin';
class App extends Component {
    render() {
        return (
            <Router>
                <div>
                    <Route exact path="/"
component={Main}/>
                    <Route exact path="/product"
component={Product}/>
                    <Route exact path="/company"
component={Company}/>
                    <Route exact path="/dashboard"
component={Dashboard}/>

                <Route exact path="/admin"
component={Admin}/>
                </div>
```

```
            </Router>
        );
    }
}

const mapDispatchToProps = dispatch => {
    return {
        // onTodoClick: id => {
        //  dispatch(toggleTodo(id))
        // }
    }
}

const mapStateToProps = state => {
    return {
        // todos: getVisibleTodos(state.todos,
state.visibilityFilter)
    }
}
export default connect(
    mapStateToProps,
    mapDispatchToProps
```

```
  )(App);
```

Лістінг файлу App.css

```
* {
  font-family: "Mukta", sans-serif;
  padding: 0px;
  margin: 0px;
  border: 0px;
}
body {
  background-color: #F3F3F3;
}
.boxed-container {
  max-width: 1440px;
  margin: 0 auto;
  box-shadow: 0 16px 48px #E2E8ED;
}
.body-wrap {
  background: #fff;
overflow: hidden;
  display: flex;
  flex-direction: column;
```

```
  min-height: 100vh;
}
.button {
  display: inline-flex;
  padding: 8px 12px;
  background: none;
  box-shadow: 0 2px 4px 0 rgba(0,0,0,.2);
  border-radius: 2px;

  border: none;
  text-decoration: none;
  font-size: 16px;
  font-weight: 500;
  font-family: Mukta,sans-serif;
  line-height: 1.6;
  white-space: nowrap;
  cursor: pointer;
  transition: background .2s ease-in-out;
  justify-content: center;

 align-items: center;
```

```
}
.button-primary {
  background: #5f67ff;
  background: linear-gradient(65deg, #5f67ff 0, #8086ff
100%);
}
.button-primary {
  color: #fff !important;
  transition: background .15s ease;
}
.button-shadow {
  position: relative;


}
a {
  color: inherit;
  text-decoration: underline;
}
a {
  background-color: transparent;
  -webkit-text-decoration-skip: objects;
```

```
}
.flexWrap {flex-wrap: wrap}
.fColumn {flex-direction: column}
.flex1 {flex:1;}
.flex2 {flex:2;}
.flex {
  display: flex;
}
.aiStart {
  align-items: flex-start;
}
.aiCenter {
  align-items: center;
}

.aiEnd {
  align-items: flex-end;
}
.jcCenter {
  justify-content: center;
}
```

```
.jcStart {
  justify-content: start;
}
.jcEnd {
  justify-content: flex-end;
}
.jcBetween {
  justify-content: space-between;
}
.header {
  background-color: #ffffff;
  /* box-shadow: rgba(0,0,0,0.5) -1px 0px -1px 1px; */
}
.clickable {
  user-select: none;
  cursor: pointer;
}
.colorWhite {
  color: white;
}
```

```
.grayText {
  color: #C9C9C9;
}
.fsize21 {font-size: 21px;}
.fsize32 {font-size: 32px;}
.grayText2 {
  color: #AEAEAE;
}
.headerTab {
  padding: 10px;
  height: 20px;
  margin: 0px 15px;
}
.headerTab2 {

  padding: 10px 15px;
  height: 20px;
}
.headerTabActive {
  padding: 10px;
```

```
  height: 20px;
  margin: 0px 15px;

}
.subHeader {
  background-color: white;
  height: 300px;
}
.grayWrapper {
  width: 100%;
  background-color: #F3F3F3;
}
.grayContent {
  border-radius: 4px;
  min-height: 100vh;
  /* background-color: white; */
}
.center {
  margin: 0px auto;
}
.container {
```

```
  max-width: 1280px;
}

.primary {
  background-color: #FF9B0C;
}
.secondary {
  background-color: #6ABF7A;
}
.primaryF {
  color: #FF9B0C;
}
.secondaryF {
  color: #6ABF7A;
}
.productCard {
  margin-right: 20px;
  margin-top: 20px;
  width: 300px;
  height: 420px;
  background-color: white;
```

```
  border-radius: 8px;
  display: flex;

flex-direction: column;
}
.productCard1 {
  margin-right: 20px;
  margin-top: 20px;
  width: 300px;
  height: 420px;
  background-color: white;
  border-radius: 8px;
  display: flex;
  flex-direction: column;
}
.productCard:hover, .productCard:hover .customProduct {
  background-color: rgba(0,0,0,0.15);
}
.productCardImage {
  width: 100%;
```

```
  height: 282px;
  background-size: cover;
  background-repeat: no-repeat;

 background-position: center;
}
.productCardHoverButton {

  margin-left: 20px;
  width: 60px;
  height: 60px;
  background-color: #FF9B0C;
  border-radius: 50%;
}
.backDefault {
  background-size: contain;
  background-repeat: no-repeat;
  background-position: center;
}
.productCardTitle {
  margin-left: 8px;
```

```
  color: #aeaeae;
  font-size: 21px;
  font-weight: 600;
}

.productCardDescription {
  margin-left: 8px;
}
.shevron {background-image: url(./assets/shevron.svg)}


.meat1 {background-image: url(./assets/meat1.jpg)}
.meat2 {background-image: url(./assets/meat2.jpg)}
.meat3 {background-image: url(./assets/meat3.jpg)}
.meat4 {background-image: url(./assets/meat4.jpg)}

.zlaki1 {background-image: url(./assets/zlaki1.jpg)}
.zlaki2 {background-image: url(./assets/zlaki2.jpg)}
.zlaki3 {background-image: url(./assets/zlaki3.jpg)}
.zlaki4 {background-image: url(./assets/zlaki4.jpg)}
```

```
.frukti1 {background-image: url(./assets/frukti1.jpg)}
.frukti2 {background-image: url(./assets/frukti2.jpg)}
.frukti3 {background-image: url(./assets/frukti3.jpg)}
.frukti4 {background-image: url(./assets/frukti4.jpg)}

.ovochi1 {background-image: url(./assets/ovochi1.jpg)}
.ovochi2 {background-image: url(./assets/ovochi2.jpg)}
.ovochi3 {background-image: url(./assets/ovochi3.jpg)}
.ovochi4 {background-image: url(./assets/ovochi4.jpg)}

.mtop8 {margin-top:8px}
.mtop24 {margin-top:24px}

.mbottom8 {margin-bottom:8px}
.mbottom24 {margin-bottom:24px}
.mright8 {margin-right:8px;}
.mleft8 {margin-left:8px;}
.mleft24 {margin-left:24px;}

.customProduct {
```

```
  background-color: #aeaeae;
  color: white;
  font-size: 72px;
}

.popup {

z-index: 99;
  position: fixed;
  top: 0;
  right: 0;
  left: 0;
  bottom: 0;
  background-color: rgba(0,0,0,.5)
}

.popupRect {
  background-color: white;
  padding: 72px 48px;
  max-width: 360px;
}
```

```
.nod {
  display: none;
}



.-facebook {

  color: #fff;
  width: calc(50% - 8px);
  background-color: #3b5998;
}

.button .ico {
  fill: currentColor;
}

.fillCurrent {

  fill: currentColor;
}
```

```
.field {
  padding: 26px 16px 9px;
  width: 100%;
  box-shadow: none;
  border-radius: 0;
  -webkit-appearance: none;
  -moz-appearance: none;

font-family: Roboto,sans-serif;
  line-height: 20px;
  font-size: 16px;
  background: #eee;
  transition: all .2s ease-in-out;
  border: none;
  border-bottom: 1px solid grey;
}

.form-floating {
  position: relative;
}
```

```
.form-floating label {
  position: absolute;
  left: 0;
  right: 0;
  top: 10px;
  padding: 6px 16px 0;
  font-size: 16px;
  color: grey;



  pointer-events: none;
  -webkit-transform: translateZ(0);
  transform: translateZ(0);
  -webkit-transform-origin: left top;
  transform-origin: left top;
  transition: all .18s cubic-bezier(.4,0,.2,1);
}

.-wide {
  width: 100%;
}
```

```
.kys {

  /* padding: 26px 16px 9px; */
  /* padding: 16px 16px; */
  padding-left: 16px;
  padding-right: 16px;
  color: white;
}



.relative {
  position: relative;
}

.cross {
  position: absolute;
  top: 16px;
  right: 16px;
  width: 16px;
  height: 16px;
```

```
}

.productRow {
  margin-top: 20px;

  width: 100%;
  /* height: 320px; */
  background-color: white;
  border-radius: 8px;
}

.productRowImage {
  width: 300px;
  height: 300px;
  /* border-right: 2px solid #FF9B0C; */
  background-size: contain;
  background-repeat: no-repeat;
  background-position: center;
}
```